SYNOPSYS®
Silicon to Software™

ASIP Designerで実現する RISC-Vプロセッサの開発と既存RTLを用いた機能拡張

RISC-V Days Tokyo 2021 Autumn

日本シノプシス合同会社
シニアアプリケーションエンジニア 伴野 充

2021年11月18日

アジェンダ

- シノプシスプロセッサソリューション
- ASIP Designerのご紹介
- RISC-Vプロセッサの開発
- 既存RTLを用いた機能拡張
- まとめ

© 2021 Synopsys, Inc.

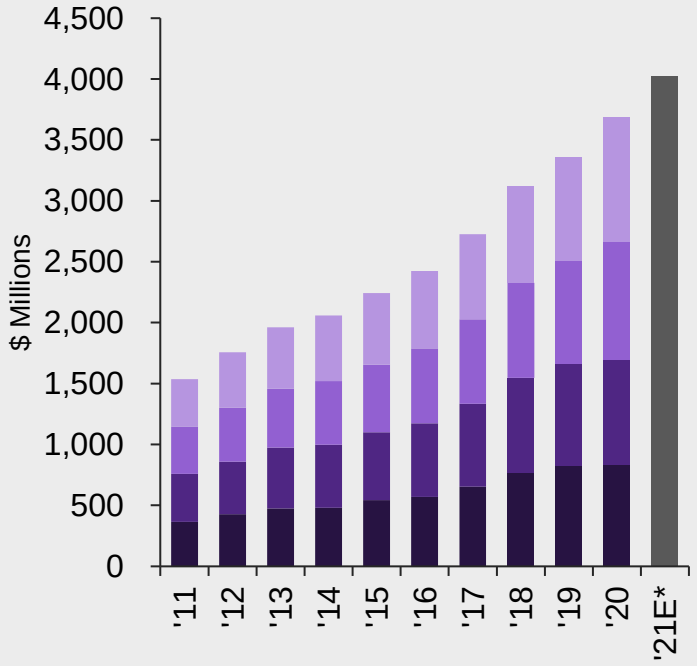
Synopsys Today: シリコンからソフトウェアへ

FY20 売り上げ:
~$3.7B

従業員数:
~15,000

取得特許:
~3,300

オフィス拠点:
~125



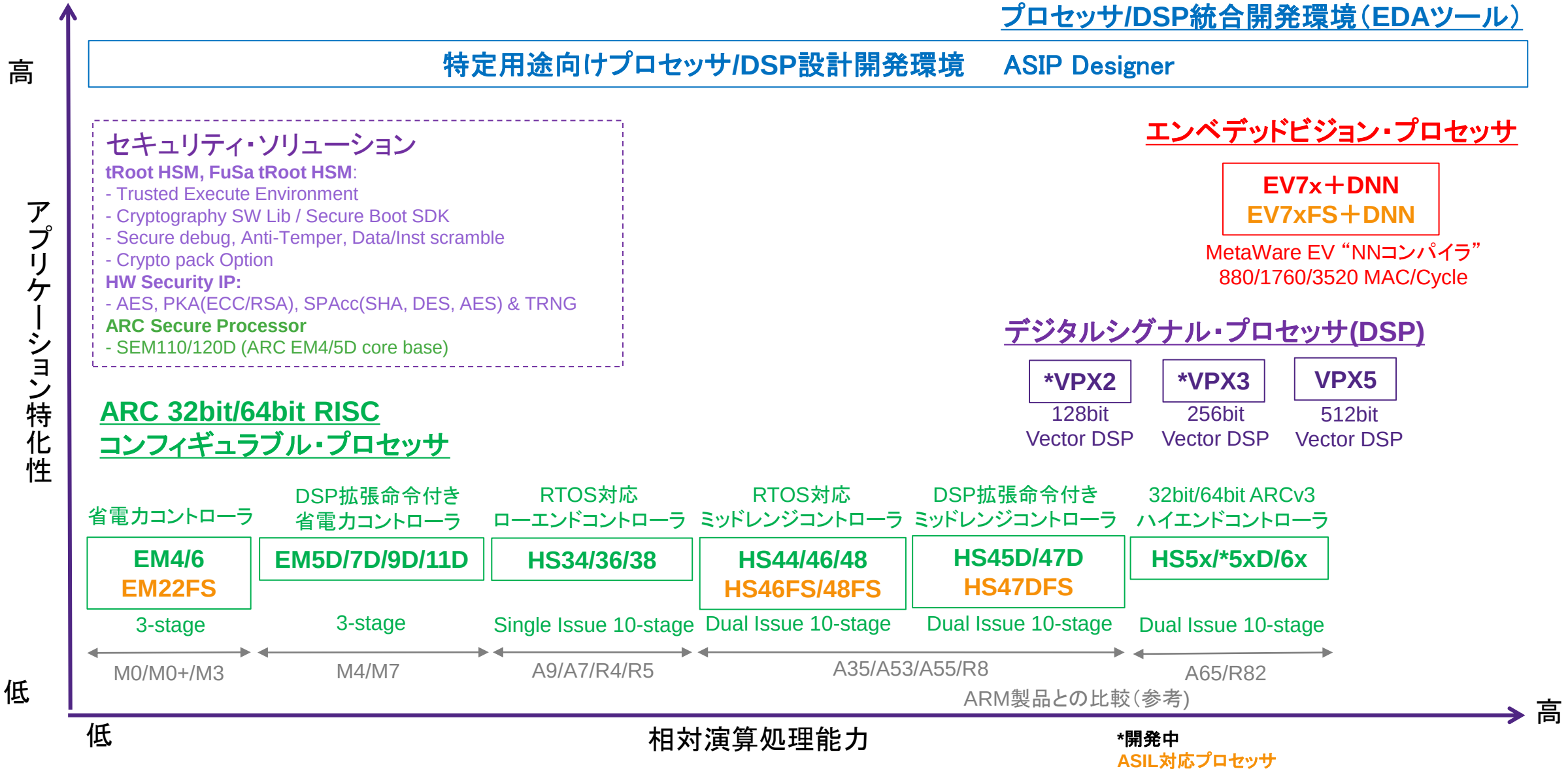
業界ナンバーワンの（EDA)電子設計自動化ツールとサービス

幅広いIPポートフォリオと#1のインターフェースIP、アナログ、組込みメモリ、物理IPおよびプロセッサ

ガードナーのアプリケーションセキュリティ・テスト向け“Magic Quadrant” のリーダー

© 2021 Synopsys, Inc.

シノプシス・プロセッサ製品一覧



© 2021 Synopsys, Inc.

ASIP Designerのご紹介

© 2021 Synopsys, Inc.

ドメインに特化したプロセッサの時代

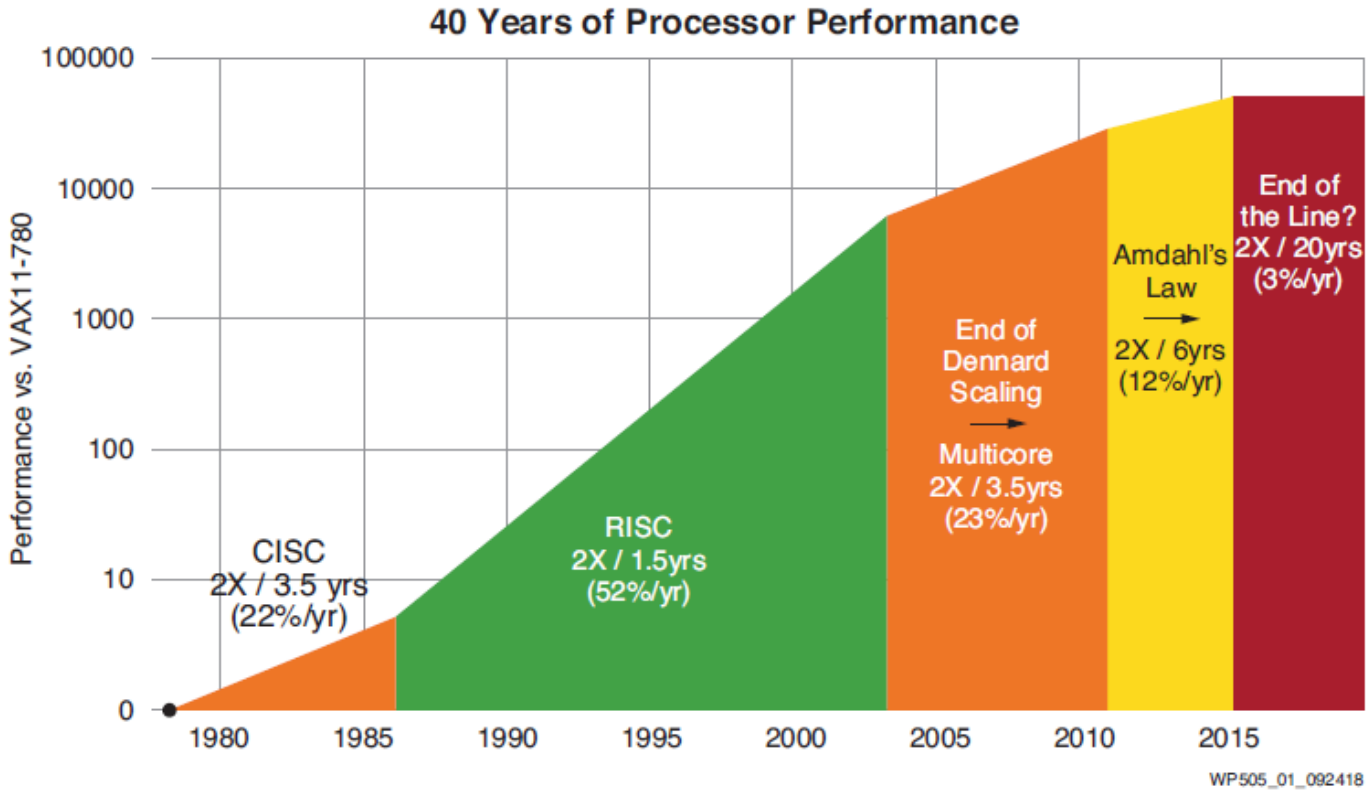


Figure 1: **Processor Performance vs. Time**

Source: Xilinx Whitepaper 505 – Versal ACAP
(reference to J. Hennessy, D. Patterson, *Computer Architecture: A Quantitative Approach* (6th Edition, 2019)

- 1995 – 2003:

  プロセス技術による性能（より高い周波数だが同じ電力消費）およびメモリアクセスの最適化（Dennard Scaling）
- 2003: Multicore

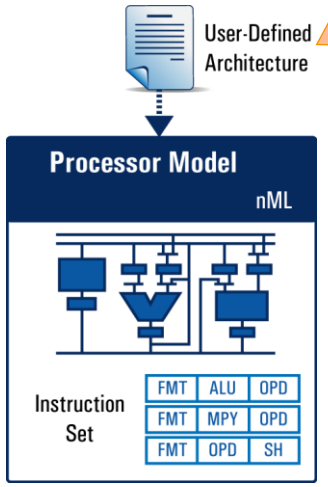
  タスクの並列化制限に到達（Amdahlの法則）
- 2015: Domain-Specific Processors

  ヘテロジニアス・マルチコア設計への移行

© 2021 Synopsys, Inc.

ASIP Designer

- アーキテクチャ構造記述言語(nML)により、
  - **アプリケーションに特化した命令セットを持つ専用プロセッサ/ DSP を自動生成する設計環境**



nML記述

```
action {
  stage E1:
    vecd =
     vsub(vecr=V[r],vecs=V[t])@vec;
    V[t] = vect = vabs(vecd) @vabs;
  }
  syntax : "vadiff v"t ",v"r ",v"t;
  image  : t::r;
```

ASIP設計開発ツール
“ASIP Designer”

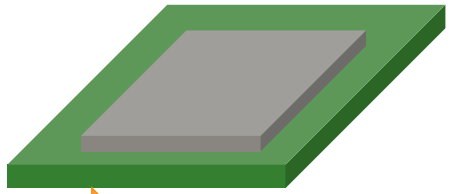
命令セット
シミュレータ(ISS)

SystemC ベース
System Simulator

C/C++ コンパイラ
アセンブラ/リンカー

デバッガ/プロファイラ

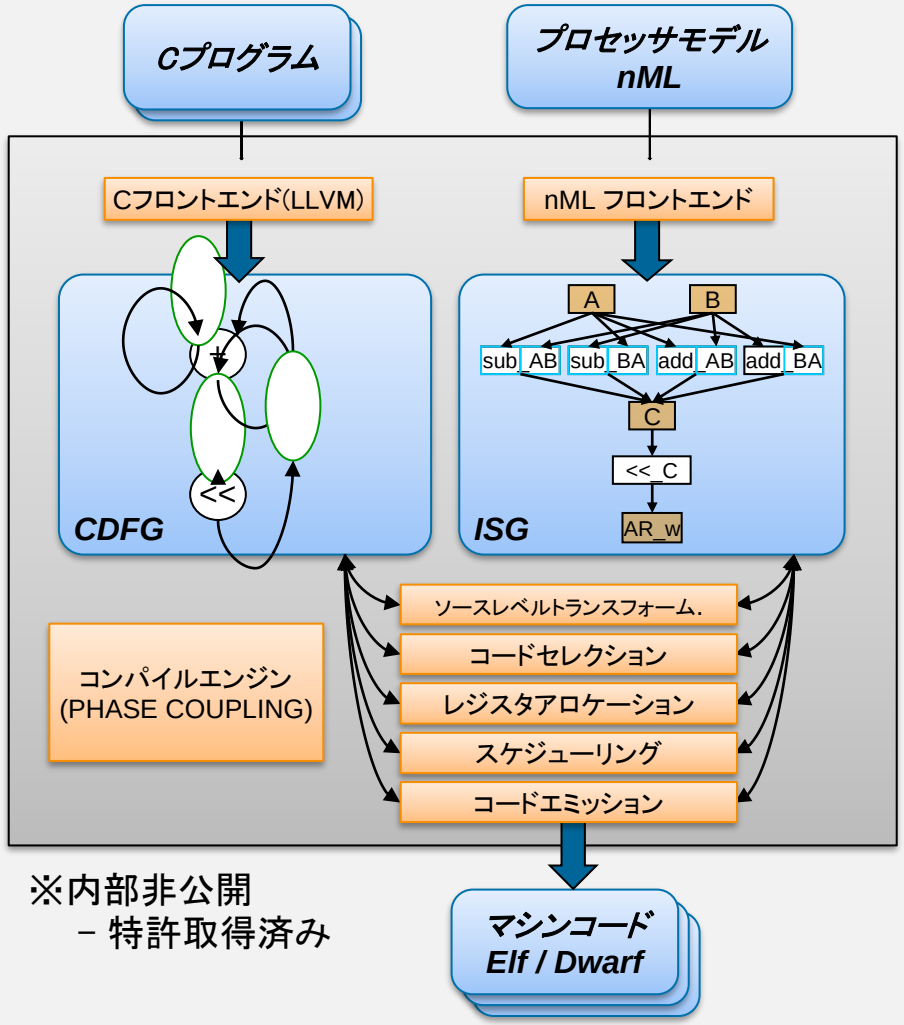
Verilog / VHDL
RTL コード



ASIC / FPGA

© 2021 Synopsys, Inc.

Graph-Based C コンパイラ “Chess”

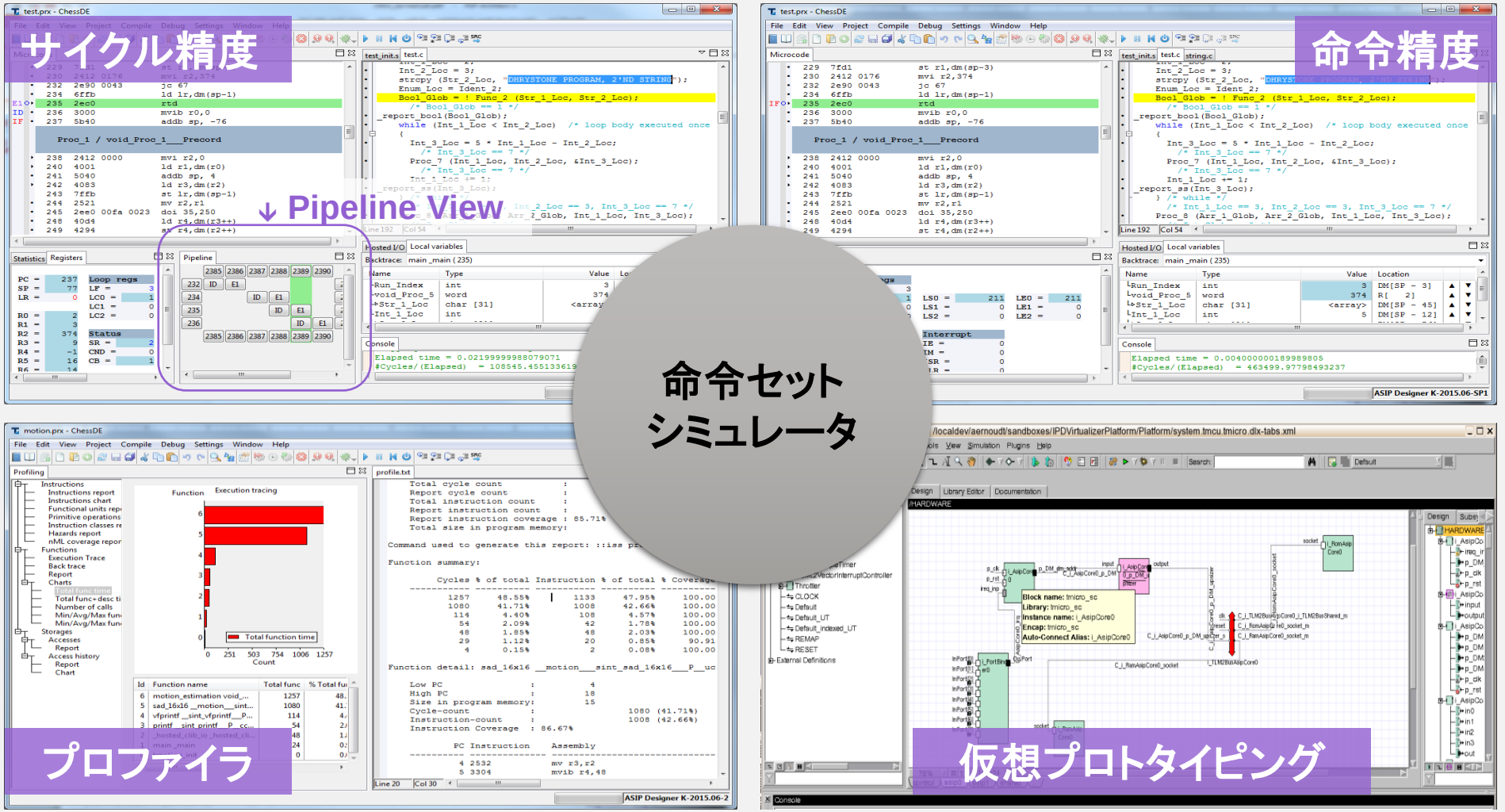


※内部非公開
– 特許取得済み

- フロントエンド
  - C → コントロールデータフローグラフ
  - nML → インストラクションセットグラフ
- コンパイルフェーズ
  - CDFGから ISGへマッピング
  - 独自“Graph”アルゴリズム
- ISGに含まれる構成情報
  - ハードウェアリソース
  - データタイプ
  - 接続情報
  - 命令エンコード情報
  - 命令レベルの並列化情報
  - 命令パイプライン情報

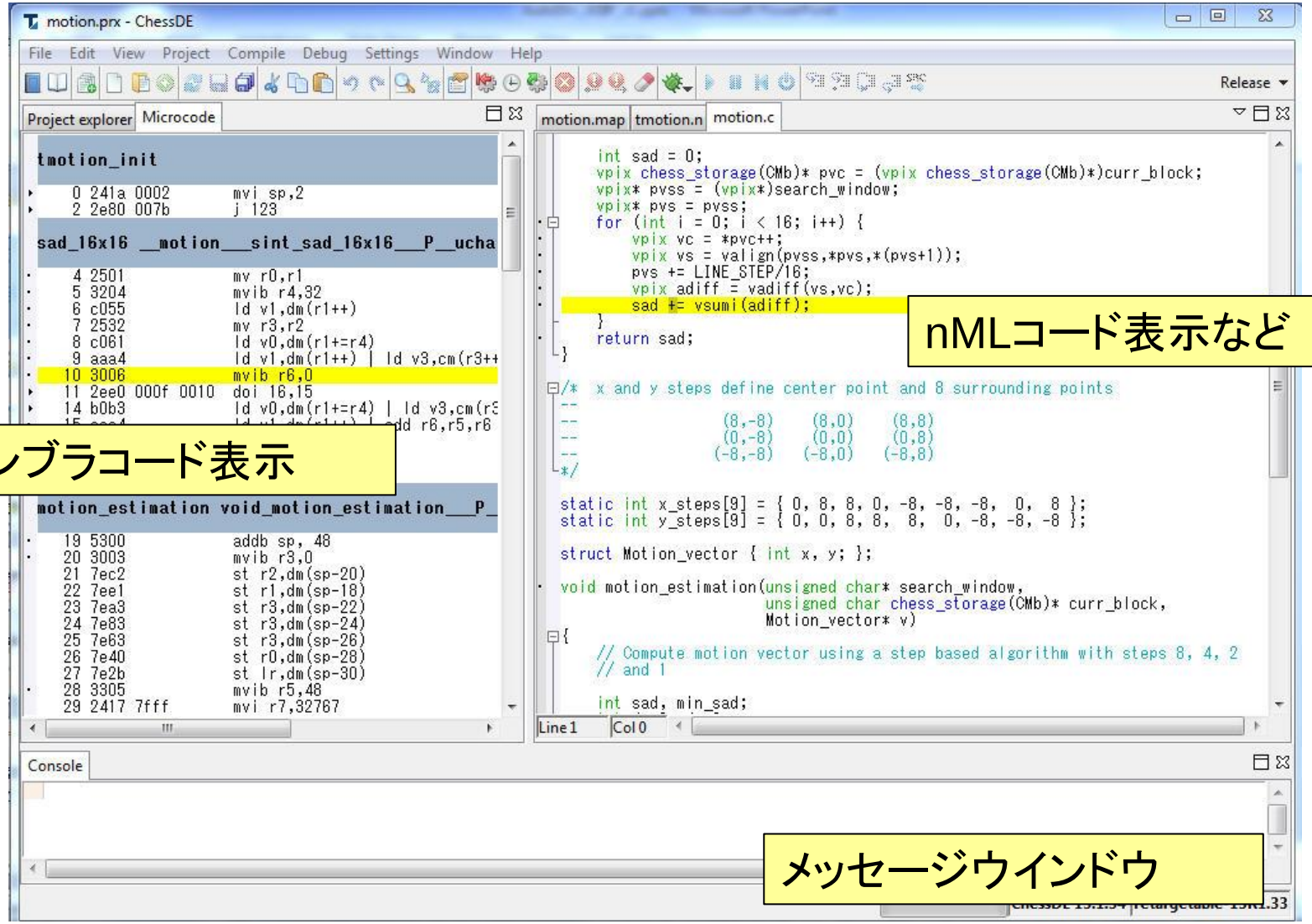
© 2021 Synopsys, Inc.

ASIP Designer - 命令セットシミュレータ



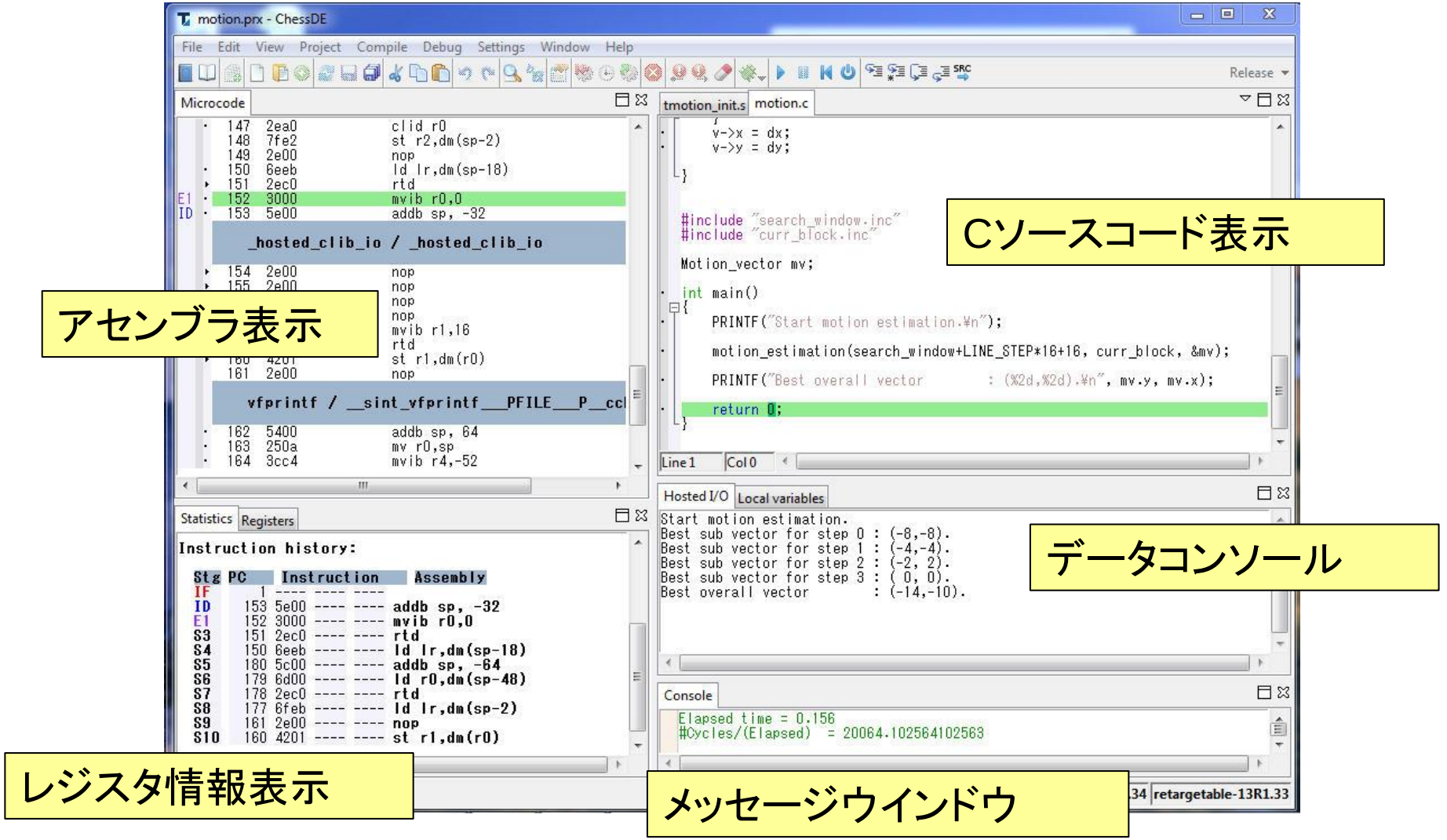
© 2021 Synopsys, Inc.

ASIP Designer生成 ー ASIPデバッガ



© 2021 Synopsys, Inc.

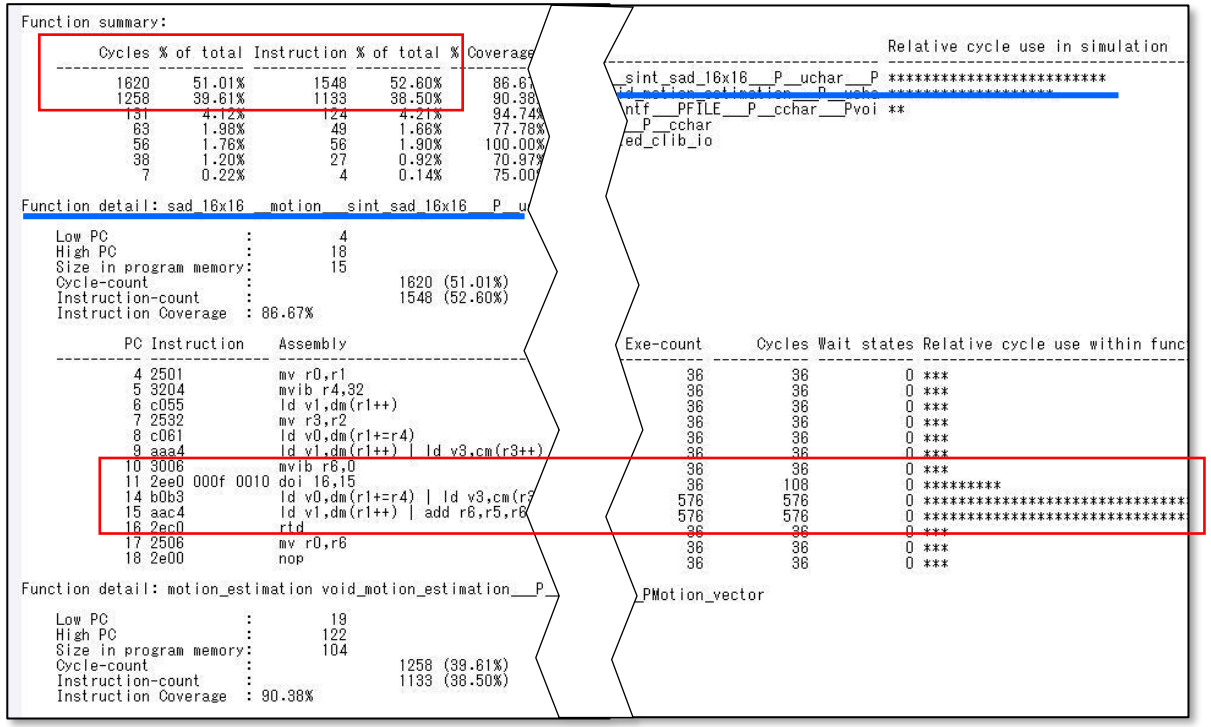
ASIP Designer生成 ー ソフトウェアデバッガ



© 2021 Synopsys, Inc.

ソフトウェア・プロファイリング機能

命令プロファイリング



```
Function summary:

      Cycles % of total Instruction % of total % Coverage
------------ ---------- ----------- ---------- ----------
        1620     51.01%        1548     52.60%     86.67%
        1258     39.61%        1133     38.50%     90.38%
         131      4.12%         124      4.21%     94.74%
          63      1.98%          49      1.66%     77.78%
          56      1.76%          56      1.90%    100.00%
          38      1.20%          27      0.92%     70.97%
           7      0.22%           4      0.14%     75.00%

Function detail: sad_16x16__motion__sint_sad_16x16__P_u

    Low PC                :          4
    High PC               :         18
    Size in program memory:         15
    Cycle-count           :                  1620 (51.01%)
    Instruction-count     :                  1548 (52.60%)
    Instruction Coverage  : 86.67%

           PC Instruction    Assembly
---------- ---------- ----------------------------------------
            4 2501           mv r0,r1
            5 3204           mvib r4,32
            6 c055           ld v1,dm(r1++)
            7 2532           mv r3,r2
            8 c061           ld v0,dm(r1+=r4)
            9 aaa4           ld v1,dm(r1++) | ld v3,cm(r3++)
           10 3006           mvib r6,0
           11 2ee0 000f 0010 doi 16,15
           14 b0b3           ld v0,dm(r1+=r4) | ld v3,cm(r3
           15 aac4           ld v1,dm(r1++) | add r6,r5,r6
           16 2ec0           rtd
           17 2506           mv r0,r6
           18 2e00           nop

Function detail: motion_estimation void_motion_estimation___P_

    Low PC                :         19
    High PC               :        122
    Size in program memory:        104
    Cycle-count           :                  1258 (39.61%)
    Instruction-count     :                  1133 (38.50%)
    Instruction Coverage  : 90.38%

                                                  Relative cycle use in simulation
------------------------------------------------ ------------------------------
_sint_sad_16x16___P__uchar___P **************************
_motion_estimation___P__uche **********************
ntff___PFILE___P__cchar___Pvoi **
__P__cchar
ed_clib_io

Exe-count      Cycles Wait states Relative cycle use within func
--------- ---------- ----------- ------------------------------
       36         36           0 ***
       36         36           0 ***
       36         36           0 ***
       36         36           0 ***
       36         36           0 ***
       36         36           0 ***
       36         36           0 ***
       36        108           0 *********
      576        576           0 ********************************
      576        576           0 ********************************
       36         36           0 ***
       36         36           0 ***
       36         36           0 ***

_PMotion_vector
```

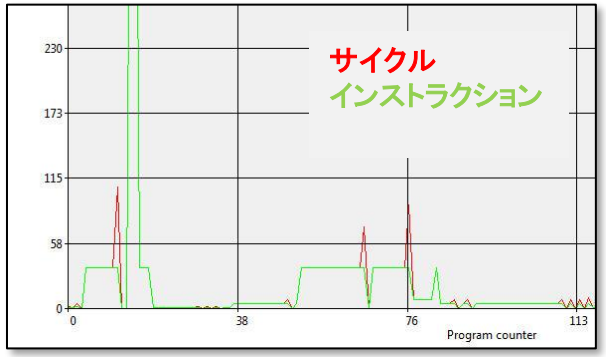
命令＆サイクル・カウント

```
ISS command      = <PROCDIR>/../iss/tmotion_ca
ISS mode         = Cycle accurate

Cycle count        = 3176
Instruction count  = 2943

PC =   130

SP =    98
Stack area: DMb[    2 ..  2049], growing up
Maximum stack pointer value =   146

Instruction history:

 Stg PC   Instruction    Assembly
 IF   130 ---- ---- ----
 ID   180 5c00 ---- ---- addb sp, -64
 E1   179 6d00 ---- ---- ld r0,dm(sp-48)
 S3   178 2ec0 ---- ---- rtd
 S4   177 6feb ---- ---- ld lr,dm(sp-2)
 S5   161 2e00 ---- ---- nop
 S6   160 4201 ---- ---- st r1,dm(r0)
 S7   159 2ec0 ---- ---- rtd
 S8   158 3101 ---- ---- mvib r1,16
 S9   157 2e00 ---- ---- nop
 S10  156 2e00 ---- ---- nop
```

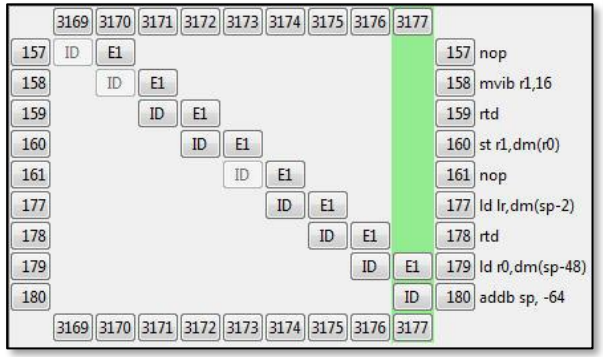
nMLカバレッジ

```
tmotion  (0.266224)
 .alu_instr  (0.0654762)
  .alu_rrr  (0.142857)
   +alu_op  (0.142857)
     -add  (1)
     -addc  (0)
     -sub  (0)
     -subb  (0)
     -and  (0)
     -or  (0)
     -xor  (0)
  .compare_rr  (0.25)
   +compare_op  (0.25)
     -lt  (1)
     -ltu  (0)
     -le  (0)
     -leu  (0)
     -gt  (0)
     -gtu  (0)
     -ge  (1)
     -geu  (0)
  .equal_rr  (0)
  .alu_rr  (0)
  .minmax_rrr  (0)
  .select_rrr  (0)
 .shift_instr  (0.333333)
  .shift_rrr  (0.333333)
```

命令＆サイクル・チャート

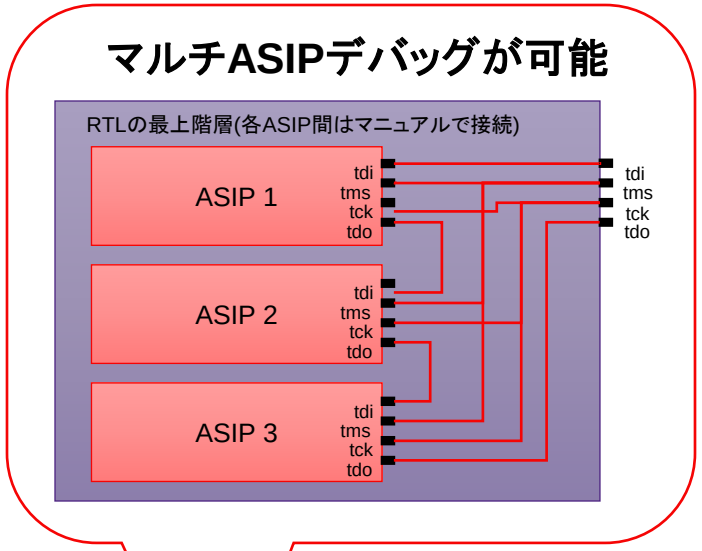


パイプラインビュー

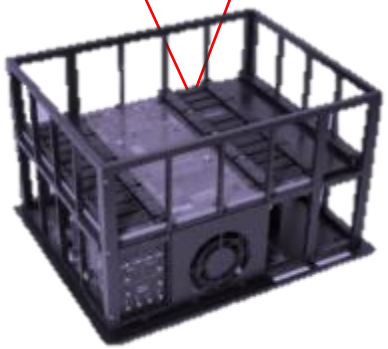


© 2021 Synopsys, Inc.

RTLコード生成と実機デバッグ環境の構築



- nML(PDG)記述からRTLコードを生成
  - Verilog/VHDL RTLコード
  - JTAG Key/JTAG key2をサポート
  - Design Compiler/ Simplify用スクリプト
  - **生成プロセッサ（RTL）のロイヤリティ不要**



HAPS-80 FPGAボード
シリーズ

Xilinx USB2

Digilent JTAG HS2

Macraigor

Lauterbach

FTDI USB Hi-Speed to MPSSE cable.
Segger J-Link Ultra cable.

ソフトウェア
デバッガ

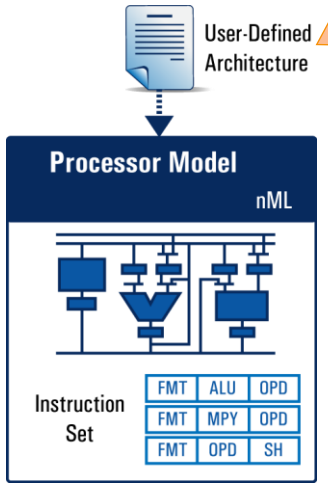
© 2021 Synopsys, Inc.

RISC-Vプロセッサの開発

© 2021 Synopsys, Inc.

ASIP Designer

- アーキテクチャ構造記述言語(nML)により、
  - **アプリケーションに特化した命令セットを持つ専用プロセッサ/ DSP を自動生成する設計環境**



nML記述

```
action {
  stage E1:
    vecd =
     vsub(vecr=V[r],vecs=V[t])@vec;
    V[t] = vect = vabs(vecd) @vabs;
  }
  syntax : "vadiff v"t ",v"r ",v"t;
  image  : t::r;
```

ASIP設計開発ツール
“ASIP Designer”

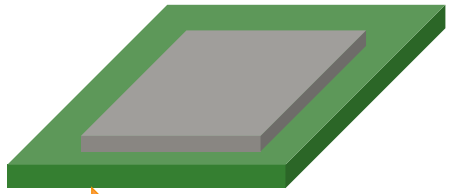
命令セット
シミュレータ(ISS)

SystemC ベース
System Simulator

C/C++ コンパイラ
アセンブラ/リンカー

デバッガ/プロファイラ

Verilog / VHDL
RTL コード



ASIC / FPGA

© 2021 Synopsys, Inc.

ご提供可能なExampleプロセッサ

種類	説明
マイクロコントローラ	
Tnano	16-bit microcontroller, lightweight and configurable
Tmicro	16-bit microcontroller, fully featured
DLX (family)	Variants of 32-bit microcontroller
Tmcu	32-bit microcontroller
Trv32 (family)	32-bit microcontroller featuring RISC-V ISA, 3 or 5 pipeline stages
Trv64 (family)	64-bit microcontroller featuring RISC-V ISA, 3 or 5 pipeline stages
PD_Triop	32-bit microcontroller with 64-bit address spaces
DSP及び並列演算エンジン	
Tdsp	16/32-bit DSP
Tvec (family)	Variants of SIMD processor
Tvliw (family)	Variants of VLIW processor
暗号処理エンジン	
Tsec	High throughput SHA/RSA/AES accelerator
Tcript	High throughput AES accelerator
アプリケーション特化	
Tmoby	Accelerator for AI application, featuring MobileNet V3 graph
Tvox	Accelerator for SLAM (simultaneous localization and mapping)
MMSE	Minimum Mean Square Error Equalization
Tgauss	Vectorization and memory management for image processing
Tmotion	Accelerator of motion estimation kernel
Tcom8	SIMD processor optimized for some communication kernels
FFTcore	Scalar implementation of complex FFT

green = released in 2020

nML記述 ソースコード ご提供

豊富なExample

- マイクロコントローラ
- DSPs
- SIMD、VLIW
- マルチスレッド
- セキュリティ（AES、SHA等）
- メモリI/F、AXIバス等

Example使用のメリット

- 基本的な機能は既に設計開発済み
- 動作可能な例を参照可能
- モデリングの概念を効率的に習得
- nMLソースコードで提供されるため、ユーザーが自由に追加変更可能

© 2021 Synopsys, Inc.

Trv(RISC-V ISA)モデル

内容

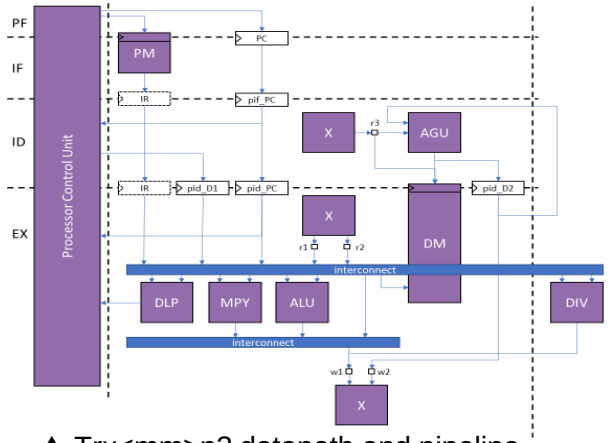
- RISC-V ISAに最適化されたモデル
- ISA拡張のスターティングポイントとしての活用 (ASIP設計)

	32-bit データパス	64-bit データパス
3ステージ パイプライン	Trv32p3 Trv32p3x	Trv64p3 Trv64p3x
5ステージ パイプライン	Trv32p5 Trv32p5x	Trv64p5 Trv64p5x

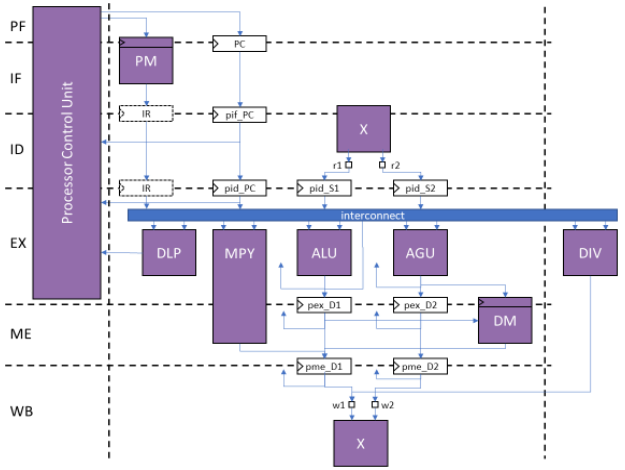
機能 (2020.03)

- 対応ISA: RV64IM, RV32IM
  - 整数命令（Integer）
  - 乗算命令（Multiply）
- オプション拡張: Trv<mm>p<n>x
  - (標準) 圧縮命令(Compressed：IMC)
  - (カスタム) ゼロオーバヘッドループ
  - (カスタム) アドレス更新後のロード/ストア

- マイクロアーキテクチャの特長
  - 5段ステージパイプライン: IF, ID, EX, ME, WB
  - 3段ステージパイプライン: IF, ID, EX
  - 保護されたパイプライン（Protected pipeline）
    - 可能な場合はレジスタをバイパス
    - バイパスが出来ない時はStall (bubble)
  - ハードウェア乗算器
    - 64x64→128→select 64bit, 32x32 → 64 → select 32bit
  - 繰り返し処理の除算器



▲ Trx<mm>p3 datapath and pipeline



▲ Trx<mm>p5 datapath and pipeline

© 2021 Synopsys, Inc.

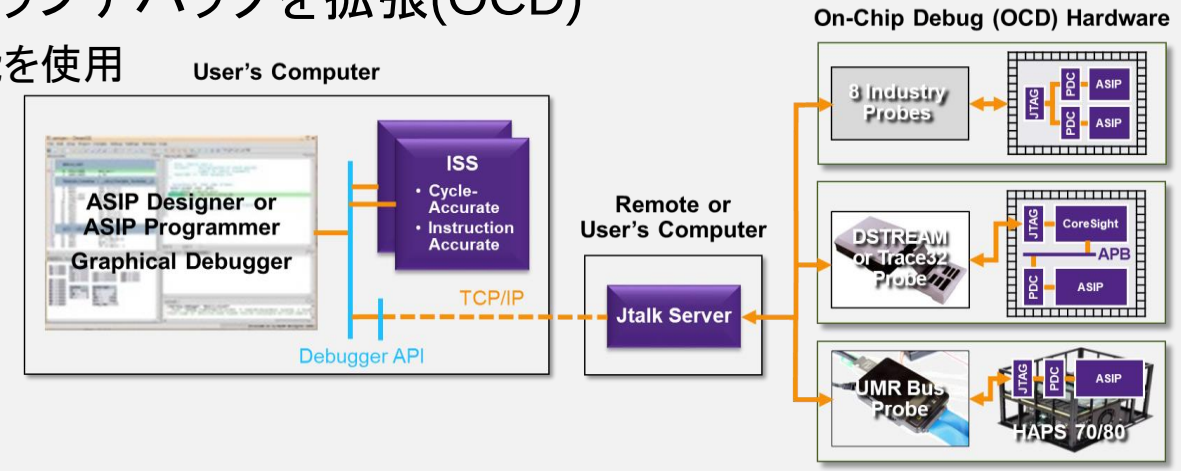
Trvモデル拡張 (2021.03)

F-クラスサポート, オンチップデバッグ

- Trv32モデルにオプションのF命令クラスを拡張
  F命令クラス
  - IEEE754 単精度浮動小数点のサポート
    (round to nearest-even mode)
  - Fレジスタファイル
  - FADD, FSUB, FMUL, FCMP (EQ/LT/LE), FMIN, FMAX, FCVT → DesignWareモジュール
  - FDIV, FSQRT → 繰り返し処理の実装

ISA	28nm TSMC	
	HS (maxfreq)	HD (minarea)
Trv32p5f (IMF)	1.37 GHz 75.6 kGE	50MHz (target) 54.7 kGE

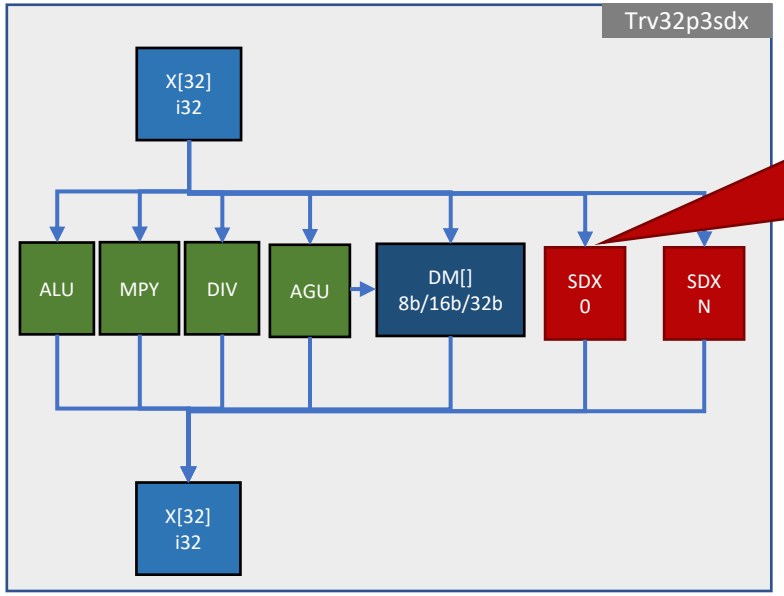
- Trvモデルにオプションのオンチップデバッグを拡張(OCD)
  - ASIP DesignerのOCD自動生成機能を使用
  - 自動的にISAの拡張に適応



© 2021 Synopsys, Inc.

SDX: シンプルデータパス拡張

- SDXは（RISC-V）プロセッサモデルに単純な拡張命令を追加するメカニズム
  - 拡張命令スケルトンは、プロセッサモデルの一部として提供
    - ユーザーは特別なnML記述を学ぶ必要がない
  - ユーザーが拡張機能の動作をPDGとしてコーディング
    - PDGは、データパス関数の動作を定義するために使用されるCのような言語
  - 拡張命令をターゲットとするコンパイラ組み込み関数を提供
- RISC-Vの“custom-2” 命令としてエンコード（後述）
- 命令バリエーション
  - 3つのreg 32ビットおよび64ビット、累積、追加のシングルレジスタ入力および出力等々, …
- 例: SHA256 (オペレーション融合), FFT (コンプレックス処理), CNN (パッケージ化されたSIMD)



Behavioral model (PDG)

```
w32 sdx0(w32 a, w32 b)
{
  w32 r;
  r[15:0] = a[15:0] + b[15:0];
  r[31:16] = a[31:16] + b[31:16];
  return r;
}
```

Compiler intrinsics

```
int sdx0(int,int);
int add2(int,int);
```


© 2021 Synopsys, Inc.

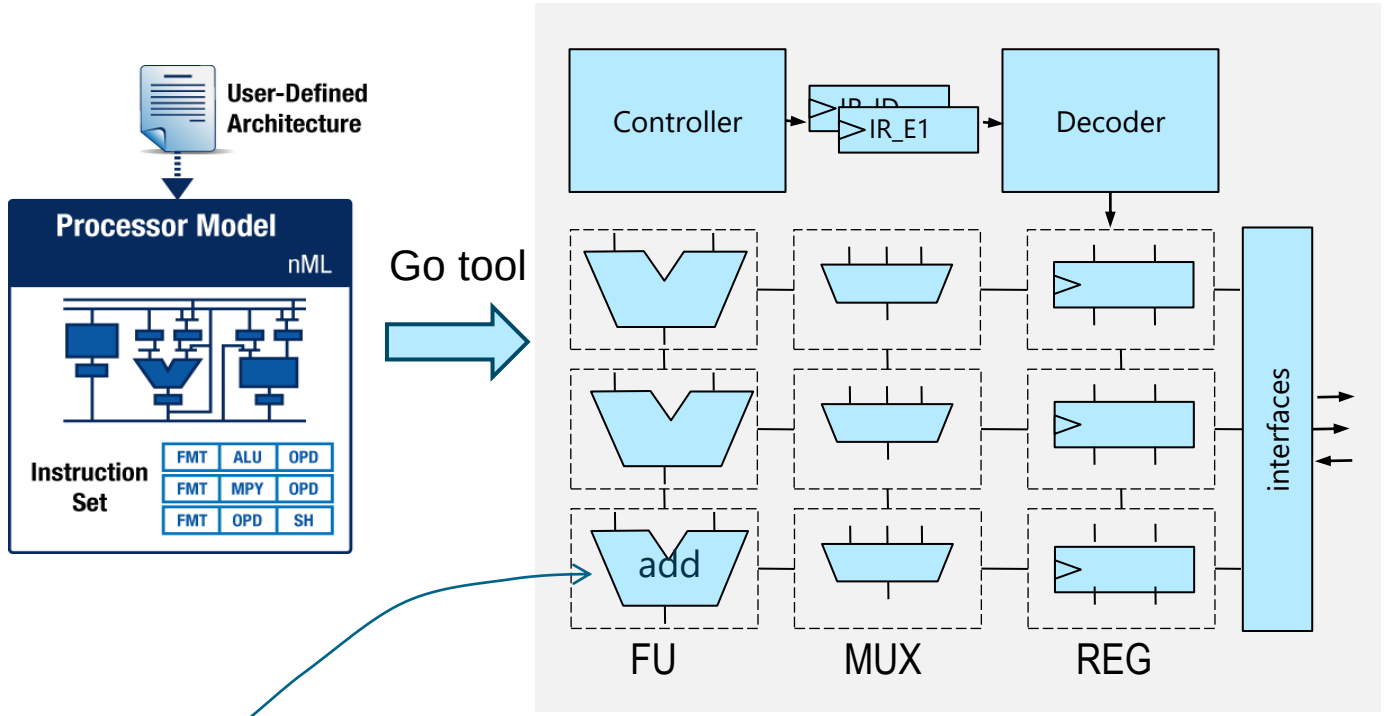
既存RTLを用いた機能拡張

© 2021 Synopsys, Inc.

ASIP DesignerにおけるRTL生成

- 合成可能なVHDL/Verilogコードを生成
- 生成されたコードの構成
  - トップレベルネットリスト
    - ファンクショナルユニット(FU)、レジスタ、およびその他のモジュール
    - MUXベースのインターコネクト
  - モジュールのビヘイビアモデル
- FUのビヘイビアモデルはPDG言語で記述
  - 例: add with status output

```
word add(word a, word b, astat& s)
{
  int17_t aa = (uint16_t)a;
  int17_t bb = (uint16_t)b;
  int17_t d = aa + bb;
  s = (d[16]^d[15])::(d<0)::(d==0);
  return d;
}
```



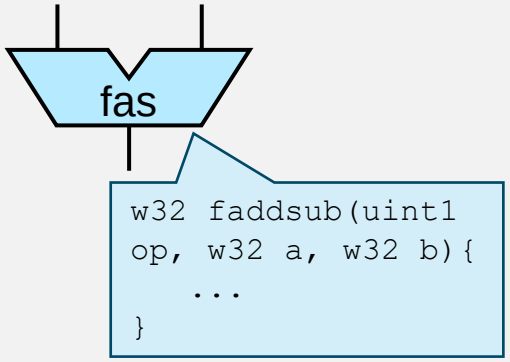
© 2021 Synopsys, Inc.

ケース1: 既存RTLによる組合せ回路の実装

FLX

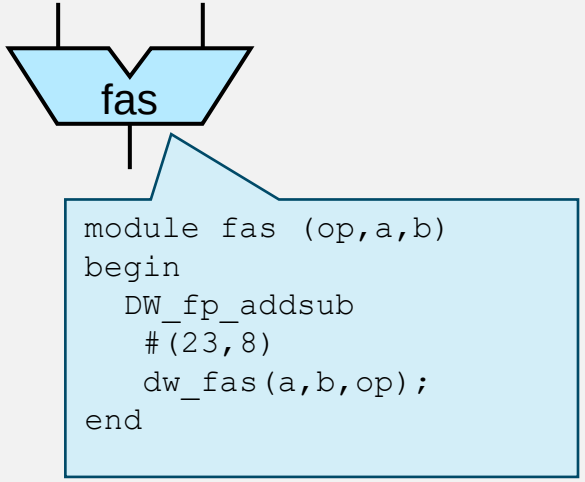
- ハードウェアによる浮動小数点をサポートした32-bit RISC
- 異なるFPU実装による3つの派生形

FLX1: PDGベースモデル



- IEEE単精度浮動小数点を実装
- PDGモデルを使用
  - ISSモデル
  - RTL生成

FLX2: DesignWareベースのRTL



- RTL実装: DWインスタンスを持つラッパー
- ISSモデル: C++ビルトイン浮動小数点加算

FLX3: Non-IEEEフォーマット

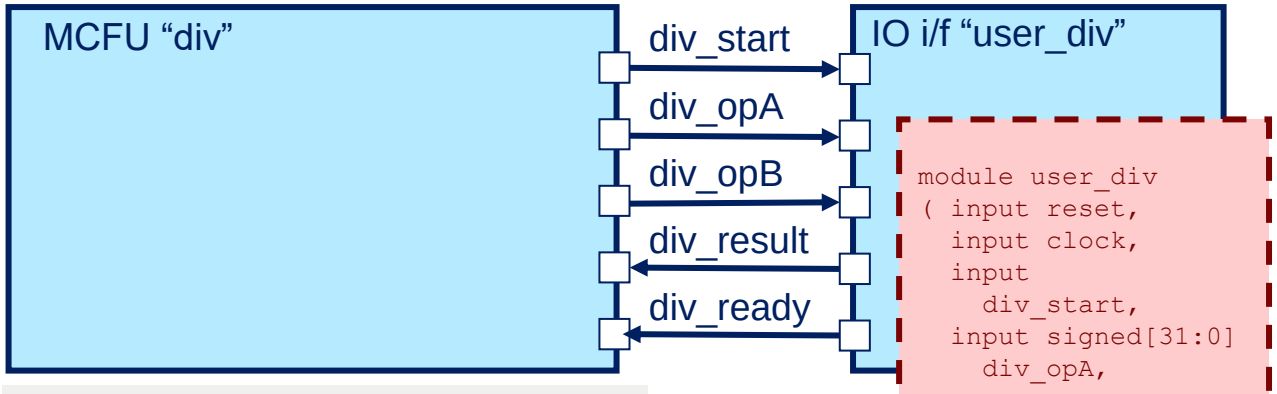
```
class float24_t
   property(
      24 bits,
      sign_magnitude,
      8 exp_bits);
```

- RTL: DesignWare
- ISS & Native
  - リファレンスのVerilogモデルをVerilatorツールでC++に変換

© 2021 Synopsys, Inc.

ケース2: 既存RTLによる繰返し操作の実装

既存のVerilog RTLモジュールとISSを用いてマルチサイクル除算を実装




```
process set_inputs() {
  div_start = vd_divide_...EX_sig;
  if (vd_divide_...EX_sig) {
    div_opA = divA;
    div_opB = divB;
  }
}
process read_outputs() {
  uint64_t result = div_result;
  if (div_ready) {
    PL = wl = result[31:0];
    PH = wh = result[63:32];
  }
}
process state_to_core() { ... }
```

```
module user_div
( input reset,
  input clock,
  input
    div_start,
  input signed[31:0]
    div_opA,
  …
)
```

user_div.v

```
class user_div {
...
}

user_div::main() {
  ...
  mdl->...->eval();
  ...
}
```

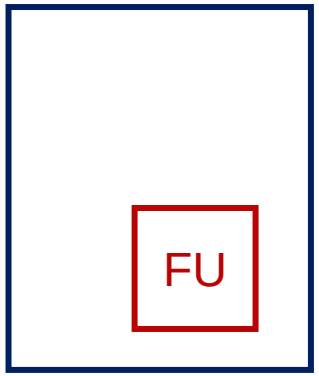
user_hdl.cpp

- 除算はマルチサイクルのプリミティブとしてモデル化
- **マルチサイクルファンクショナルユニット**を使用して、
  - Start信号をドライブして、オペランドを出力ポートdiv_opAと div_opBにコピー
  - ready信号を確認して、結果をライトバック
  - パイプライン制御用の信号を生成
- **IOインターフェース**をRTLモジュールに対するプレースホルダーとして使用される
  - GoツールがVerilogファイルにあるuser_div.vをIOインターフェースモジュールの代わりにコピー
- **IOインターフェース**はISSに対してuser_div C++クラスを生成するためにも使用される
  - コンストラクタ `user_div(dlx*)`
  - メンバ `main()`
  - ステータスアップデート `update_status()`
  - Verilatorツールによってuser_div.vをビットアキュレートかつサイクルアキュレートなC++コードに変換

© 2021 Synopsys, Inc.

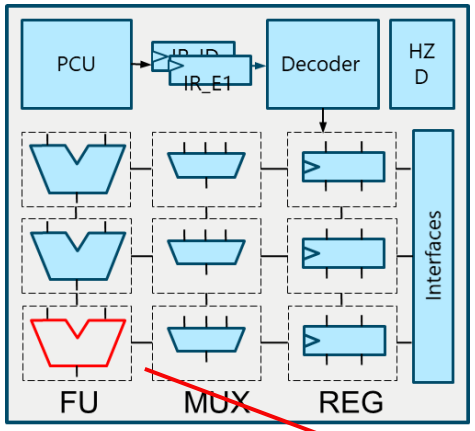
既存RTLコードのインポートフロー

PDGで定義されていない
ダミーのファンクショナル
ユニットを持つnMLモデル



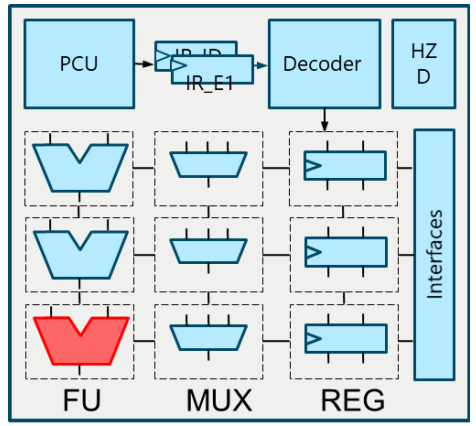
初回のGo
ツールの実行

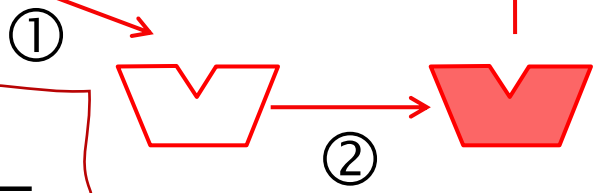
空のファンクショナルユニッ
トを持つ生成されたRTL



2回目のGo
ツールの実行、
既存RTLをイン
ポート

インポートされたファンクショナル
ユニットを持つ再生成されたRTL





ラッパーの作成
① 空モジュールを持つファイルをコピー
② 既存RTLコードを統合
③ 編集したモジュールをGoツールのコン
フィギュレーションオプションでインポート

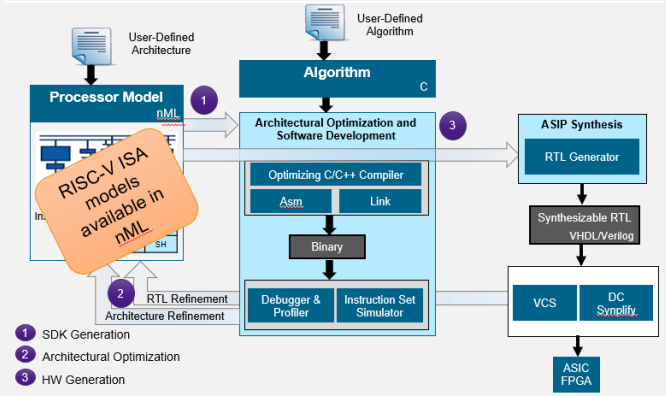
© 2021 Synopsys, Inc.

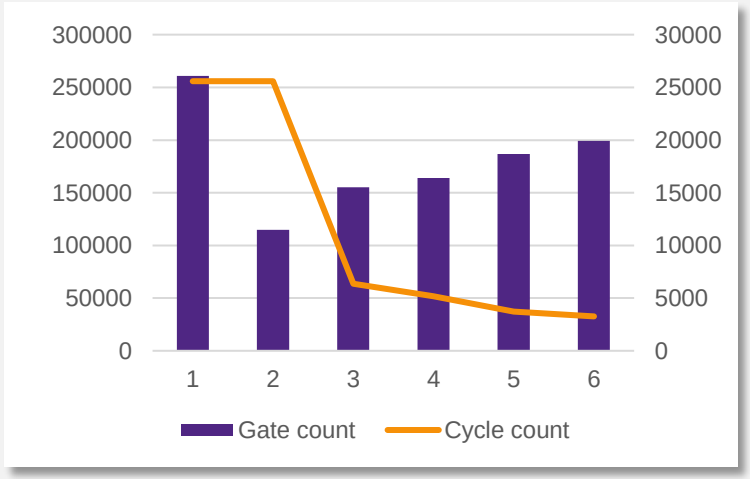
まとめ

© 2021 Synopsys, Inc.

ASIP Designer – ASIP デザインを容易に

- アプリケーション特化によるPPAの大幅な改善！
- RISC-V ISAベースのExampleモデルで即座にトライアルが可能！
- 既存の設計資産(RTL)を即座に流用可能！
- **製品の差別化を実現する拡張可能かつロイヤリティフリーなRISC-Vプロセッサを開発！**
  - ご興味のある方はぜひご連絡ください
  - コンタクト先: jp-mkg_info@synopsys.com





© 2021 Synopsys, Inc.

SYNOPSYS®
Silicon to Software™

Thank You